Z^2 traversal order: An interleaving approach for VR stereo rendering on tile-based GPUs

Jae-Ho Nah¹, Yeongkyu Lim¹ (✉), Sunho Ki¹, and Chulho Shin¹

© The Author(s) 2017. This article is published with open access at Springerlink.com

Abstract With increasing demands of virtual reality (VR) applications, efficient VR rendering techniques are becoming essential. Because VR stereo rendering has increased computational costs to separately render views for the left and right eyes, to reduce the rendering cost in VR applications, we present a novel traversal order for tile-based mobile GPU architectures: Z^2 traversal order. In tile-based mobile GPU architectures, a tile traversal order that maximizes spatial locality can increase GPU cache efficiency. For VR applications, our approach improves upon the traditional Z order curve. We render corresponding screen tiles in left and right views in turn, or simultaneously, and as a result, we can exploit spatial adjacency of the two tiles. To evaluate our approach, we conducted a trace-driven hardware simulation using Mesa and a hardware simulator. Our experimental results show that Z^2 traversal order can reduce external memory bandwidth requirements and increase rendering performance.

Keywords virtual reality (VR); tile traversal order; tile-based GPU; mobile GPU; graphics hardware

1 Introduction

Recent progress in head-mounted displays (HMDs) and GPUs has brought an explosion in the virtual reality (VR) market. As a result, a wide range of VR applications has been developed: games, 360°

video, simulations, social media, and so on. In these VR applications, immersive visual experience is very important. Thus, VR devices usually need to provide a high resolution screen (up to UHD) with high refresh rates (up to 120 Hz). Additionally, head-mounted displays (HMDs) require separately rendered images for left and right eyes, and this stereo VR rendering can require up to twice the number of drawing calls. Therefore, efficient VR rendering techniques are required for realistic VR experiences.

Among various approaches to accelerate VR rendering, we focus on efficient GPU hardware architectures for VR applications. Currently most mobile GPUs (e.g., Qualcomm Adreno, ARM Mali, and Imagination Technologies PowerVR) are based on tile-based GPU architectures to minimize off-chip memory accesses. These architectures divide the entire screen into multiple tiles by adding a tiling stage between vertex and fragment shading. Each shader core performs fragment shading of geometry in each screen tile using a tile buffer. Because the number of tiles is much higher than the number of shader cores, different tile traversal orders exist, and there is a need for efficient tile traversal orders which improve rendering performance by utilizing spatial locality. There is a further chance to improve VR rendering performance by choosing a tile traversal order specifically designed for VR applications.

In this paper, we present a novel tile traversal order called the Z^2 traversal order. This traversal order is based on the traditional Z order curve (also known as Morton order) [1], but we improve upon it for VR stereo rendering in two ways. The left-right tile assignment version (Z^2 LRTA) assigns tiles in the left and right screens to a shader core

¹ LG Electronics, 19, Yangjae-daero 11-gil, Seocho-gu, Seoul, Republic of Korea. E-mail: J.-H. Nah, nahjaeho@gmail.com; Y. Lim, postrain70@gmail.com (✉); S. Ki, sunho.ki@lge.com; C. Shin, chulho.shin@lge.com.

Manuscript received: 2017-04-11; accepted: 2017-07-12

in turn, while the simultaneous tile access version (Z^2 STA) assigns corresponding tiles in the left and right screens to a shader core simultaneously, performing interleaved access to the primitive lists of the two tiles. Thanks to similarity of the left and right screens, Z^2 traversal order can increase spatial locality compared to traditional traversal orders. From the point of view of simultaneous rendering of the left and right views, the idea of our traversal order was inspired by Hasselgren et al.'s multi-view rasterization architecture [2]. However, there is a difference between them: Hasselgren et al. focused on how to efficiently rasterize triangles for multiple views, while we focus on how to efficiently map shader cores to screen tiles.

We have built a GPU simulation environment using Mesa 11.0.3 [3] to evaluate the effectiveness of the Z^2 traversal order. This simulation environment provides an analysis of texture cache access patterns. This analysis includes texture cache hit rates, memory bandwidth requirements for texture mapping, and the utilization of a texture mapping unit on the simulator. Our experimental results using various scenes: GFXBench [4] T-Rex, Crytek Sponza, and L-Bench [5] Sponza (see Fig. 1), show that Z^2 traversal order can reduce memory bandwidth requirements for texture mapping by up to 48% and can increase texturing performance by up to 9% compared to the traditional Z order curve.

This paper is an extended version of our previous conference paper presented in ACM SIGGRAPH Asia 2016 Technical Briefs [6]. We extend our previous work by providing a more detailed description of our simulation environments, using a more accurate experimental setup and additional scenes, providing new experimental results and analyses, and discussing limitations of our approach.

2 Background and related work

2.1 Tile-based GPU architectures and tile traversal order

Tile-based GPU architectures have been widely adopted for use in bandwidth-limited mobile platforms. GPU microarchitectures are differently implemented by each GPU vendor, but those using tile-based architectures have a common feature: splitting the entire rendering stage by dividing the screen into small tiles and redistributing primitives into multiple shader cores using the primitive list in each tile. This redistribution is performed between vertex and pixel processing, so the tile-based GPU architectures are also known as sort-middle architectures [7]. Because depth and color accesses for fragment shading can be performed using a small tile buffer in such architectures, they reduce the number of power-hungry DRAM accesses and so are suitable for mobile devices. An excellent overview of tile-based rendering is provided in Harris [8].

After the tiling stage, tiles and their primitive lists are distributed on multiple shader cores. If a tile traversal order is cache friendly, it can increase rendering performance, as GPU architectures usually include multi-level cache hierarchies. Scanline order is the simplest form, the Z order curve [1, 9] is a more sophisticated form which increases spatial locality, and a zig-zag pattern [10] is an alternative to the Z order curve. Figure 2 illustrates examples of these three tile traversal orders.

2.2 VR acceleration techniques

A brute-force approach for VR stereo rendering is to separately render scenes twice for the left and right views. This approach is simple but doubles the number of drawing calls. Thus, recent work

Fig. 1 VR images captured in various scenes: GFXBench T-Rex (left), Crytek Sponza (middle), and L-Bench Sponza (right). By exploiting similarity between the images of the left and right views, our novel tile traversal order decreases memory bandwidth requirement for texture mapping by up to 48% and increases texturing performance by up to 9% when rendering these scenes. A barrel distortion correction shader was not applied in our experiments as its texture access pattern is scene independent.

Fig. 2 Examples of tile traversal orders: scanline order (left), Z order curve (middle), and zig-zag order (right).

tries to reduce the redundant CPU/GPU workload in that case. An alternative approach is shader multi-view (also known as stereo instancing) [11–13]; by exposing a ViewID variable to shaders, a GPU can separately handle shader threads for each view without increased draw-call overhead. This method can be implemented on current generation GPUs with recent OpenGL/OpenGL ES extensions (GL_OVR_multiview2, GL_NV_viewport_array2, and GL_ARB_shader_viewport_layer_array). A more aggressive approach is shading reuse [2, 11]; by reusing fragment shading results from the left view for the right view, fragment shading costs can be reduced by up to half. However, this can degrade image quality because pixel values in the right view are approximately evaluated on the texture space; this is particularly problematic for view-dependent shading. Another approach is to broadcast drawing calls across multiple GPUs [11, 14–16]. This approach can utilize the full power of multiple GPUs connected by SLI or Crossfire interfaces.

Another research direction for VR rendering is to reduce the number of shaded fragments of each view. Vlachos [17] presented a stencil mesh to cull hidden areas after warping in advance. Foveated rendering [18] is a gaze-contingent multi-resolution rendering technique. By using eye trackers, this technique lowers image quality in the periphery (outside the fovea) to increase rendering performance. The image quality of the peripheral area can be improved by a radially progressive blur, post-process contrast enhancement, and temporal anti-aliasing [19]. NVIDIA multi-resolution shading [16] allows multiple scaled viewports in a single pass, and as a result, the edges of the screen distorted by warping and lens distortion can be rendered at reduced resolution without apparent loss of image quality. It can also be used for fixed foveated rendering [14].

3 Z^2 tile traversal order

In this section, we describe our novel Z^2 tile traversal order. The most important point of tile traversal orders is how much they increase spatial locality; if data from similar texture addresses are referenced again within a short period of time, there will be a high possibility of retaining the texture data in cache hierarchies.

We observe that images of the left and right views in VR stereo rendering usually look similar as illustrated in Fig. 1. This is because the same scene is rendered from two slightly different viewpoints. Therefore, if we are able to render two screen tiles in the left and right views in turn or simultaneously, this will increase cache locality. Our Z^2 traversal order utilizes this approach for VR stereo rendering. Apart from this difference, the traversal order in each view is fundamentally the same as the Z order curve.

We introduce two different traversal orders as depicted in Fig. 3: the left-right tile assignment version (Z^2 LRTA) and the simultaneous tile access version (Z^2 STA). Z^2 LRTA traverses the tiles in the left and right screens in turn. When multiple shader cores share an L2 cache, this traversal order can increase L2 cache hit rates by assigning different shader cores to the left and right screen tiles respectively. This traversal order can be simply implemented on Z -curve-order-based architectures without increase in hardware complexity.

In contrast to Z^2 LRTA, Z^2 STA fetches two tiles, for left and right screens, simultaneously, so two triangle lists for the two tiles are passed to a single shader core. Next, triangles in the left and right triangle lists are rendered in turn; in other words, after a triangle in the left screen tile is rendered, a triangle in the right screen tile is rendered. If the two screen tiles consist of very similar triangles (e.g., regions in the far distance), Z^2 STA can increase not only spatial locality but also temporal locality because there is a high possibility that the same triangle is fetched again as the two tiles are rendered. However, this traversal order has a disadvantage compared to Z^2 LRTA; Z^2 STA requires double-sized working sets to render two screen tiles concurrently. Thus, to support Z^2 STA, the tile memory size needs to be doubled or the tile size needs to be halved. The former option can decrease area efficiency due to increased tile buffer size. However, the portion

Fig. 3 Traversal order examples for 32 tiles in VR stereo rendering (top) and tile assignment examples for the tiles using a two-core GPU (bottom).

of a tile buffer in an entire GPU is usually small; for 128-bit per-pixel data and a tile size of 16×16 , only a 4 KB tile buffer is needed per core [20]. If we additionally consider a 32-bit depth/stencil buffer, the total tile buffer size is 5 KB. Even if the tile buffer size is larger to support multi-sampled anti-aliasing (MSAA) or large color formats, it is unlikely that a double-sized tile buffer makes a large increase in the entire GPU area. Of course, a smaller tile size is more practical because it does not require any additional area. However, smaller tile sizes may increase tiling overheads if there are many triangles in a scene.

Note that both Z² STA and Z² LRTA require the use of shader multi-view techniques and multi-view extensions mentioned in Section 2.2. If a brute-force approach using duplicated drawing calls is used, there is no clue how to obtain the geometry lists of the left and right views simultaneously. Thus, our traversal order can be enabled only if the tiling stage can sort all geometry for the left and right views together in a single frame.

4 Results and discussion

4.1 Experimental setup

We have built a texture-mapping simulation environment using Mesa 11.0.3 [3] to verify the effectiveness of Z² order—see Fig. 4. When an OpenGL application with a VR library is executed, the Mesa OpenGL library is used. For software rendering, the Gallium Softpipe driver in Mesa is called instead of the actual GPU driver. Mesa as modified by us generates texture access traces, and our in-house hardware simulator uses the trace files to calculate statistics.

Fig. 4 Simulation environment.

We now give implementation details. As Mesa is a software OpenGL renderer based on immediate-mode rendering, we may modify its texturing pipeline to emulate memory access patterns on tile-based GPU architectures. First, we make a 16×16 virtual grid because, usually, at least a size of 16×16 is used in modern tile-based GPUs (e.g., ARM Mali). Next, we store texture addresses into the corresponding grid cells whenever texture accesses occur. As a result, each grid cell has texture addresses per tile per render pass per frame. After a frame is rendered, the texture trace files generated by traversing the tiles with the methods in Fig. 3 are fed to our in-house hardware simulator in order to measure cache hit/miss rates, memory bandwidth requirements, and utilization of the texture mapping unit (TMU). Before performing a cache simulation, we set several parameters for the multi-level cache hierarchy, such as the cache size, cache block size, set associativity, and miss penalty for each cache memory. Cache hits and misses are accumulated during simulation. After simulation is finished, off-chip memory bandwidth requirements are calculated by multiplying average L2 miss rates by the L2 cache block size (the L2 cache is the last-level cache in our

experiments). Cache replacements are based on the least recently used (LRU) policy and performed after the miss penalty cycles of each cache. Additionally, four consecutive memory accesses are amortized when bilinear filtering is enabled.

In our TMU simulation, looping for the next chance technique [21] is adopted for hiding memory latency; when a cache miss occurs in a GPU thread, the thread is inactivated for the remaining pipeline stages, then the thread is reactivated when the thread is input to the first pipeline stage again in order to get the texel data from the cache. As described in Lee et al. [22], similar techniques are used in other commodity hardware as well. The miss penalty for L1 and L2 caches is 20 and 200 cycles, respectively.

For hardware configuration, we assume that two GPU shader cores share an L2 cache as illustrated in Fig. 3. We believe this configuration is reasonable because usually two to four GPU cores are connected with an L2 cache in modern tile-based GPUs (e.g., Mali T600–T800 series and PowerVR 6–7 series). The size of each L1 cache is set to 8 and 16 KB, and the L2 cache size is set to 128 and 256 KB. Additionally, both caches are configured to have four-way set associativity with a 16-byte block size; as the size of compressed texture data is usually 8 or 16 bytes (e.g., for DXTC, ETC, and ASTC), we set the cache block size to 16 bytes.

4.2 Experimental results

Our benchmark scenes are GFXBench [4] T-Rex, Crytek Sponza, and L-Bench [5] Sponza as depicted in Fig. 1. For VR stereo rendering, we used the Oculus VR library with a 100-degree field-of-view (FOV). All prestored textures in the scenes were compressed by DXT1 with mipmaps. The screen resolution of each view is 960×1080.

Table 1 shows that our benchmark scenes have different characteristics in terms of texture complexity. First, the T-Rex scene has similar graphics quality to that found in modern mobile games; it consists of various types of objects with multiple rendering effects. Thus, texture accesses in the scene are balanced between general pre-stored static images and framebuffer-attachable dynamic images (e.g., shadow maps). Second, Crytek Sponza is an improved, remodeled version of the original Sponza model. Compared to the original model,

Table 1 Texture complexity of benchmark scenes

Scene	Texture complexity	
	Static	Dynamic
GFXBench T-Rex	Medium	Medium
Crytek Sponza	High	N/A
L-Bench Sponza (no shadows)	Low	N/A
L-Bench Sponza (with shadows)	Low	High

Crytek Sponza has much more complex geometry and textures; in particular, most materials in the scene have two to four textures (diffuse, specular, alpha, and normal maps), so multi-texturing, which results in high static texture complexity, is necessary to render the scene. Third, the Sponza scene in L-Bench was designed to test shadow mapping performance; this scene has an additional moving hand object inside the Sponza palace for dynamic shadows, and the cascaded shadow mapping technique was implemented in the scene with a 4 × 2k texture array. To test the two extreme cases, we enabled and disabled shadow mapping. Without shadows, the scene requires only accesses to simple textures. In contrast, when we enable shadows, the majority of memory traffic is caused by accessing a large, uncompressed, non-mipmapped shadow map array because other texture data in the scene are relatively simple. Using the above four cases, we believe that our experimental setup covers a wide range of texture complexity.

Table 2 summarizes the experimental results. It includes L1 cache miss rates, L2 cache miss rates, and TMU utilization. Figure 5 compares external memory bandwidth requirements of each tile traversal order. Note that the values in Fig. 5 represent only bandwidth requirements for texture mapping because a tile traversal order is not directly related to off-chip memory bandwidth for fetching other types of data.

The results in Table 2 and Fig. 5 show that scene characteristics and cache configurations put a different complexion on the effectiveness of our approach. Our approach is effective when mip-mapping is properly accessed and many objects share identical textures. In this case, our approach can increase TMU utilization by up to 9.1% (Crytek Sponza) and can decrease memory traffic for texture mapping by up to 47.8% (L-Bench Sponza without shadows). If overall GPU performance is bounded by texture mapping performance in a texture-heavy

Table 2 Experimental results. The Z order curve (Morton), the Z^2 LRTA order, and the Z^2 STA order are compared in this table. Lower cache miss rates and higher TMU utilization are better results

Cache size (KB)		L1 cache miss rate (%)			L2 cache miss rate (%)			TMU utilization (%)		
L1	L2	Morton	Z^2 LRTA	Z^2 STA	Morton	Z^2 LRTA	Z^2 STA	Morton	Z^2 LRTA	Z^2 STA
GFXBench T-Rex										
8	128	7.65	6.86	6.70	52.07	41.49	39.72	76.75 (1.0 $\times$)	80.31 (1.046 $\times$)	80.99 (1.055 $\times$)
16	128	6.53	5.63	5.41	60.89	50.51	48.92	77.07 (1.0 $\times$)	80.79 (1.048 $\times$)	81.55 (1.058 $\times$)
8	256	6.68	6.08	6.00	35.45	25.09	24.14	82.05 (1.0 $\times$)	85.82 (1.046 $\times$)	86.21 (1.051 $\times$)
16	256	5.52	4.80	4.68	42.97	31.80	30.94	82.53 (1.0 $\times$)	86.47 (1.048 $\times$)	86.93 (1.053 $\times$)
Crytek Sponza										
8	128	16.14	15.85	15.81	61.23	47.79	48.41	66.12 (1.0 $\times$)	68.66 (1.038 $\times$)	68.56 (1.037 $\times$)
16	128	12.16	10.18	10.40	80.82	66.98	69.93	66.60 (1.0 $\times$)	70.56 (1.059 $\times$)	69.86 (1.049 $\times$)
8	256	15.95	15.61	15.57	53.02	33.14	34.44	67.70 (1.0 $\times$)	72.29 (1.069 $\times$)	71.96 (1.064 $\times$)
16	256	11.26	8.76	8.76	74.45	54.81	56.14	68.29 (1.0 $\times$)	74.51 (1.091 $\times$)	74.27 (1.088 $\times$)
L-Bench Sponza (without shadows)										
8	128	3.60	2.55	2.52	51.70	40.40	38.31	85.54 (1.0 $\times$)	90.43 (1.057 $\times$)	90.83 (1.062 $\times$)
16	128	3.16	2.10	2.04	58.84	49.47	47.76	85.77 (1.0 $\times$)	90.74 (1.058 $\times$)	91.12 (1.062 $\times$)
8	256	3.23	2.42	2.40	39.37	33.52	31.88	88.66 (1.0 $\times$)	91.85 (1.036 $\times$)	92.16 (1.039 $\times$)
16	256	2.77	1.96	1.91	45.90	41.44	40.33	88.92 (1.0 $\times$)	92.18 (1.037 $\times$)	92.49 (1.040 $\times$)
L-Bench Sponza (with shadows)										
8	128	36.43	36.38	36.83	74.31	75.31	74.60	57.57 (1.0 $\times$)	57.51 (0.999 $\times$)	57.48 (0.998 $\times$)
16	128	36.30	36.26	36.69	74.59	75.56	74.72	57.58 (1.0 $\times$)	57.51 (0.999 $\times$)	57.50 (0.999 $\times$)
8	256	35.25	35.32	35.61	36.45	40.15	34.81	62.44 (1.0 $\times$)	61.74 (0.989 $\times$)	62.68 (1.004 $\times$)
16	256	35.12	35.19	35.48	36.63	40.33	34.99	62.44 (1.0 $\times$)	61.74 (0.989 $\times$)	62.68 (1.004 $\times$)

Fig. 5 Bandwidth comparison for the traditional Z order curve (Morton), Z^2 LRTA order, and Z^2 STA order. Lower values are better.

scene, then we expect a similar rendering speedup to the increase in TMU utilization. However, if large uncompressed textures are mapped, then texels for

the two tiles fetched together from the left and right screens belong to different cache blocks, as in the L-Bench Sponza with shadows scene. In this case,

our approach does not show any advantages. If more effective rendering techniques, such as ASTC-compressed shadow maps [23], are used, we believe that memory traffic caused by shadow mapping (and other render-to-texture techniques) can be reduced, benefiting our approach.

Even though Z^2 LRTA shows better results in some cases, Z^2 STA achieves higher TMU utilization and lower memory traffic on average. Z^2 LRTA and Z^2 STA increase TMU utilization by 3.8% and 4.1% on average, respectively. In terms of bandwidth requirements, Z^2 LRTA and Z^2 STA decrease memory traffic by 24.8% and 27.9% on average, respectively.

4.3 Discussion

The scenes used in Section 4.1 are rendered with forward shading; deferred shading with a G-buffer [24] has not been widely used on mobile 3D graphics applications due to its memory bandwidth problem. If our approach is used with deferred shading, its efficiency would be reduced compared to the forward shading case. Because a G-buffer is stored in multiple render targets (MRTs), the majority of memory traffic in deferred shading is usually caused by G-buffer accesses. In this case, the memory bandwidth reduction rate using our approach might be relatively low because our approach does not influence G-buffer accesses. In other words, because the G-buffers of the left and right views are stored at different memory addresses, we cannot use any coherence when accessing them even if the values of the two G-buffers are similar. Fortunately, this memory bandwidth problem for G-buffers can be alleviated with novel memory traffic reduction methods, such as pixel local storage [20]. If such methods were supported in both the GPU and the VR application, we believe that our approach with deferred shading could achieve a similar level of efficiency to that with forward rendering.

As mentioned in Section 3, our approach requires use of the multi-view extensions. Thus, if the extensions are used for other purposes in addition to VR rendering, our approach may negatively affect performance. For example, Martin et al. [23] recently presented an efficient cascaded shadow mapping algorithm using the `GL_OVR_multiview` extension. In this case, our Z^2 traversal order may show worse cache efficiency than the original traversal order

when reading triangle data for each tile in the shadow casting stage because there is little coherence between the views. To avoid this situation, the GPU driver will need to cleverly detect each case by analyzing the related parameters; for instance, our traversal order can be used only if the number of views is two and the frame buffer object format is a combination of color and depth.

The optimal tile traversal order in a scene with perfect coherence between left and right views will not generate any cache misses in rendering the right view. In this ideal case, the reduction in memory bandwidth requirement will be 50%. Because our simple interleaved tile traversal orders achieve a memory-bandwidth reduction of up to 47.8% in a common game-like scene, we believe that our approach is quite efficient.

5 Conclusions and future work

We have presented two variations of the traditional Z order curve which are specially designed for VR stereo rendering. Z^2 LRTA traverses the left and right screen tiles in turn and is advantageous in terms of hardware complexity. Z^2 STA loads the two tiles simultaneously into a single shader core and can maximize cache locality. We built a simulation environment using Mesa to evaluate the proposed tile traversal order, and the experimental results show that both approaches can decrease cache miss rates and increase TMU utilization compared to the traditional Z order curve.

As future work, we would like to experiment more with different VR algorithms and hardware architectures. Disparity manipulation techniques for specific graphics effects (e.g., gloss depiction [25]) will affect coherence between left and right views, so we would like to analyze the effect of our approach in those cases. Additionally, we believe that our interleaved Z^2 traversal order can be used in not only rasterization GPUs but also ray-tracing GPUs. If camera rays are shot using the Z order curve in a ray tracer [21], our approach is applicable to use of the ray tracer for VR.

Acknowledgements

GFXBench T-Rex, Sponza, Crytek Sponza, and Hand are courtesy of Kishonti Ltd., Marko Dabrovic,

Crytek, and the Utah 3D Animation Repository, respectively. We used the Crytek Sponza scene modified by Dario Scarpa to fill a missing texture. Sam Martin gave us useful comments. We would like to appreciate the reviewers for their valuable comments.

References

- [1] Morton, G. M. *A Computer Oriented Geodetic Data Base and a New Technique in File Sequencing*. New York: International Business Machines Company, 1966.
- [2] Hasselgren, J.; Akenine-Möller, T. An efficient multi-view rasterization architecture. In: Proceedings of the 17th Eurographics Conference on Rendering Techniques, 61–72, 2006.
- [3] Paul, B.; Whitwell, K. The Mesa 3D graphics library version 11.0.3. 2015. Available at <http://www.mesa3d.org/>.
- [4] Kishonti Informatics. GFXBench 4.0. 2016. Available at <https://gfbenchmark.com/>.
- [5] Nah, J.-H.; Suh, Y.; Lim, Y. L-Bench: An Android benchmark set for low-power mobile GPUs. *Computers & Graphics* Vol. 61, 40–49, 2016.
- [6] Nah, J.-H.; Lim, Y.; Ki, S.; Shin, C. Z^2 traversal order for VR stereo rendering on tile-based mobile GPUs. In: Proceedings of the SIGGRAPH ASIA 2016 Technical Briefs, Article No. 6, 2016.
- [7] Molnar, S.; Cox, M.; Ellsworth, D.; Fuchs, H. A sorting classification of parallel rendering. *IEEE Computer Graphics and Applications* Vol. 14, No. 4, 23–32, 1994.
- [8] Harris, P. The Mali GPU: An abstract machine, part 2—tile-based rendering. 2014. Available at <https://community.arm.com/graphics/b/blog/posts/the-mali-gpu-an-abstract-machine-part-2---tile-based-rendering>.
- [9] Clarberg, P.; Toth, R.; Munkberg, J. A sort-based deferred shading architecture for decoupled sampling. *ACM Transactions on Graphics* Vol. 32, No. 4, Article No. 141, 2013.
- [10] Ellis, S.; Engh-Halstvedt, A.; Nystad, J. Graphics processing systems. US Patent 9122646 B2, 2015.
- [11] Reed, N.; Sancho, D. VR Direct: How NVIDIA technology is improving the VR experience. In: Proceedings of the Game Developer Conference, 2015.
- [12] Wilson, T. High performance stereo rendering for VR. In: Proceedings of the San Diego Virtual Reality Meetup, 2015. Available at https://docs.google.com/presentation/d/19x9XDjUvkW_9gsfsMQzt3hZbRNzi-VsoCEHOn4AercAc/mobilepresent?slide=id.p.
- [13] Johansson, M. Efficient stereoscopic rendering of building information models (BIM). *Journal of Computer Graphics Techniques* Vol. 5, No. 3, 1–17, 2016.
- [14] Vlachos, A. Advanced VR rendering performance. In: Proceedings of the Game Developer Conference, 2016.
- [15] AMD. Virtual reality with AMD LiquidVR™ technology. 2015. Available at <http://www.amd.com/enus/innovations/software-technologies/technologies-gaming/vr>.
- [16] NVIDIA. NVIDIA® VRWorks™. 2016. Available at <https://developer.nvidia.com/vrworks>.
- [17] Vlachos, A. Advanced VR rendering. In: Proceedings of the Game Developer Conference, 2015.
- [18] Guenter, B.; Finch, M.; Drucker, S.; Tan, D.; Snyder, J. Foveated 3D graphics. *ACM Transactions on Graphics* Vol. 31, No. 6, Article No. 164, 2012.
- [19] Patney, A.; Salvi, M.; Kim, J.; Kaplanyan, A.; Wyman, C.; Benty, N.; Luebke, D.; Lefohn, A. Towards foveated rendering for gaze-tracked virtual reality. *ACM Transactions on Graphics* Vol. 35, No. 6, Article No. 179, 2016.
- [20] Bjørge, M.; Martin, S.; Kakarlapudi, S.; Fredriksen, J.-H. Efficient rendering with tile local storage. In: Proceedings of the ACM SIGGRAPH 2014 Talks, Article No. 51, 2014.
- [21] Nah, J.-H.; Kwon, H.-J.; Kim, D.-S.; Jeong, C.-H.; Park, J.; Han, T.-D.; Manocha, D.; Park, W.-C. RayCore: A ray-tracing hardware architecture for mobile devices. *ACM Transactions on Graphics* Vol. 33, No. 5, Article No. 162, 2014.
- [22] Lee, W.-J.; Shin, Y.; Hwang, S. J.; Kang, S.; Yoo, J.-J.; Ryu, S. Reorder buffer: An energy-efficient multithreading architecture for hardware MIMD ray traversal. In: Proceedings of the 7th Conference on High-Performance Graphics, 21–32, 2015.
- [23] Martin, S.; Garrard, A.; Gruber, A.; Bjørge, M.; Zioma, R.; Bengel, S.; Nummelin, N. Moving mobile graphics. In: Proceedings of the ACM SIGGRAPH 2015 Courses, Article No. 18, 2015.
- [24] Saito, T.; Takahashi, T. Comprehensible rendering of 3-D shapes. *ACM SIGGRAPH Computer Graphics* Vol. 24, No. 4, 197–206, 1990.
- [25] Templin, K.; Didyk, P.; Ritschel, T.; Myszkowski, K.; Seidel, H.-P. Highlight microdisparity for improved gloss depiction. *ACM Transactions on Graphics* Vol. 31, No. 4, Article No. 92, 2012.

Jae-Ho Nah received his B.S., M.S., and Ph.D. degrees from the Department of Computer Science, Yonsei University in 2005, 2007, and 2012, respectively. Currently, he is a senior research engineer at LG Electronics. His research interests include ray tracing, rendering algorithms, and graphics hardware.

Yeongkyu Lim received his B.S. degree from Kyungpook National University in 1997 and his M.S. degree from the Department of Computer Science, Korea University in 1999. In 2013, he received his Ph.D. degree from the Department of Computer Science, Yonsei University, Seoul, Republic of Korea. He has worked at LG Electronics since 1999. His research areas are embedded systems, HCI, and mobile GPU architectures and computing.

Sunho Ki is a senior research engineer at LG Electronics. He received his B.S. and M.S. degrees from the Department of Electronic and Electrical Engineering, Hongik University in 2007 and 2009, respectively. His main research interest is in GPU/SoC architectures and rendering algorithms.

Chulho Shin is a research fellow at LG Electronics. He received his M.S. and Ph.D. degrees in computer engineering from the University of Southern California. He received his B.S. degree from the Department of Electronic Engineering, Yonsei University. His main research interests include computing platform architectures for low power SoCs, low power CPU and GPU architectures, automated

configuration optimization for SoCs, performance estimation, and high-performance VLSI implementation.

Open Access The articles published in this journal are distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

Other papers from this open access journal are available free of charge from <http://www.springer.com/journal/41095>. To submit a manuscript, please go to <https://www.editorialmanager.com/cvmj>.